Лабораторна робота № 1
Перші кроки у Platform Designer Tool

Platform Designer Tool – це інструмент, який дозволяє розробнику створювати системи на кристалі із бібліотечних компонентів, наприклад - пам’яті, інтерфейсів введення-виведення, таймерів, процесорного ядра, обробників відео/аудіо потоків, тощо.

У даному описі розглянуто роботу у Quartus Prime 18.1 Lite із платою Terasic DE1-SoC.

Джерела, за якими підготовлено інструкцію:

Introduction to the Platform Designer Tool

ftp://ftp.intel.com/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/Introduction_to_the_Qsys_Tool.pdf

Intel FPGA Academic Program Tutorials

https://software.intel.com/en-us/fpga-academic/learn/tutorials

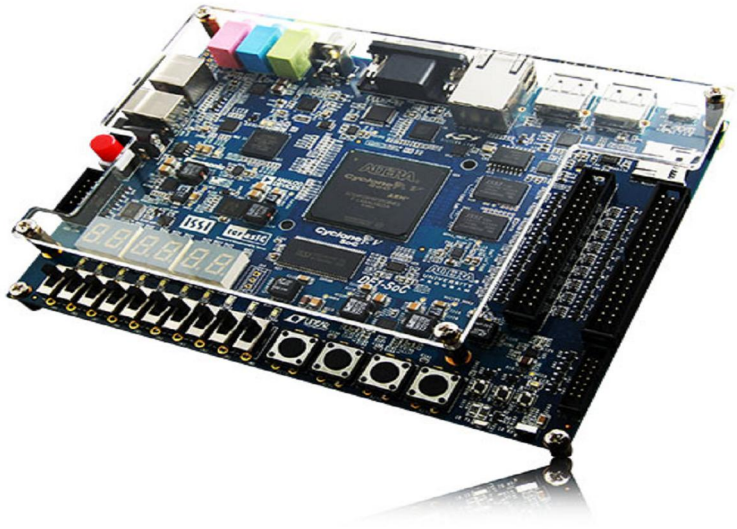
1. Вступ

Технологію розробки апаратної системи проілюстровано послідовністю покрокових інструкцій, щодо використання середовища **Platform Designer Tool** (попередня назва – **Qsys**) спільно з програмним забезпеченням **Quartus® Prime**. Останнім кроком процесу розробки є налаштування розробленого пристрою на платі з мікросхемою FPGA та запуск прикладної програми.

Малюнки в роботі були отримані з використанням середовища **Quartus Prime** версії 18.1. Для інших версій програмного забезпечення ці малюнки можуть трохи відрізнятися.

2. Плати Intel® FPGA серії DE

Для цієї роботи необхідно мати плату Intel серії DE, наприклад, таку, що показана на малюнку 1.



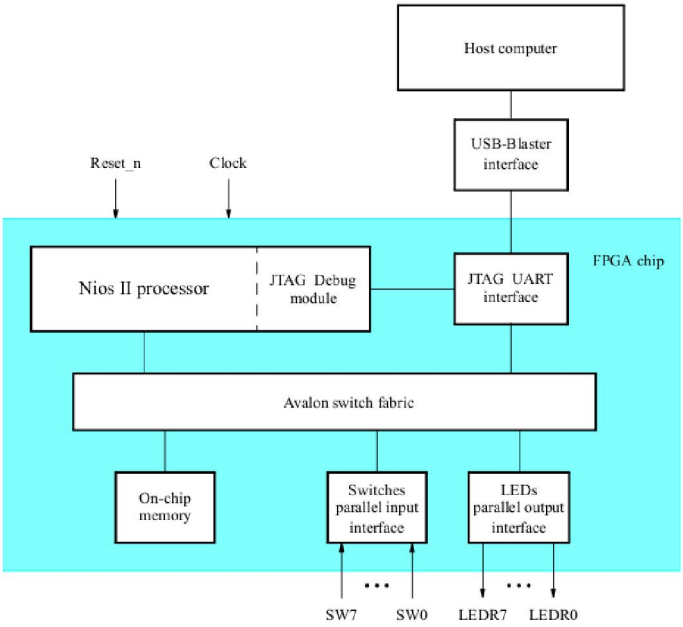
Малюнок 1. Плата DE1-SoC кампанії Intel.

На малюнку зображена плата DE1-SoC, на якій встановлено мікросхему FPGA Cyclone® V. Плата має багато додаткових

ресурсів, таких як мікросхеми пам'яті, повзунки-перемикачі, кнопки, світлодіоди, аудіовхід / вихід, відеовхід (з декодером NTSC / PAL) та відео вихід (VGA). Вона оснащена кількома типами послідовних вхідних / вихідних з'єднань, включаючи USB-порт для підключення плати до персонального комп'ютера. У цій роботі буде використано лише невелику частину ресурсів: мікросхему FPGA, повзунки-перемикачі, світлодіоди та порт USB, який підключається до комп'ютера.

3. Приклад апаратної цифрової системи

У цій роботі буде використовуватися проста апаратна система, яка показана на малюнку 2. Вона складається з вбудованого процесорного ядра Nios II, що являє собою програмний процесорний модуль, описаний мовою опису апаратури. Процесор Nios II може бути частиною більшої системи, а сама система може бути реалізована в мікросхемі Intel FPGA за допомогою програмного забезпечення Quartus Prime.



Малюнок 2. Приклад системи з ядром Nios II

Як показано на малюнку 2, процесор Nios II підключається до інтерфейсів пам'яті та вводу / виводу за допомогою системи зв'язку, яка має назву **комутатор шини Avalon**. Ця шина генерується автоматично середовищем Qsys. Компонент пам'яті в наведеної системи буде реалізовано за допомогою вбудованої пам'яті, що є в мікросхемі FPGA. Інтерфейси вводу / виводу, які підключаються до повзункових перемикачів та світлодіодів, буде реалізовано за допомогою стандартних модулів, які присутні у середовищі Qsys. Спеціальний інтерфейс JTAG UART використовується для підключення до мікросхеми, та забезпечує обмін USB-посиланнями з хост-комп'ютером, до якого підключена плата DE. Цей пристрій та його супутнє програмне забезпечення мають назву USB-Blaster.

Інший модуль, що зветься JTAG Debug, забезпечує можливість керування хост-комп'ютера процесором Nios II. Це дає можливість виконувати такі операції, як завантаження програм для Nios II в пам'ять, запуск та зупинку виконання цих програм, встановлення точок переривання та аналізу вмісту комірок пам'яті та регістрів Nios II.

Оскільки всі частини ядра Nios II, які реалізовані у мікросхемі FPGA, визначаються за допомогою мови опису апаратури, досвідчений користувач може написати такий код для реалізації будь-якої частини. Але, це було б обтяжливе і трудомістке завдання. Натомість існує можливість використання бібліотеки середовища Qsys для реалізації необхідної системи, просто вибираючи відповідні компоненти та вказуючи параметри, необхідні для пристосування кожного компонента до загальних вимог системи. Хоча, в цьому прикладі, ілюструються можливості середовища Qsys для реалізації дуже простої системи, той же підхід може використовуватися для проектування складних систем.

Система, яка зображена на малюнку 2, призначена для реалізації простого завдання. Вісім перемикачів на платі DE1-SoC, SW7 - 0, використовуються для включення або вимкнення восьми світлодіодів LEDR7 - 0. Щоб вирішити це завдання, восьмибітна послідовність, яка відповідає поточному стану перемикачів, повинна бути надіслана до вихідного порту для активації

світлодіодів. Це відбувається за допомогою програми, яка виконується процесором Nios II та зберігається в пам'яті мікросхеми. Потрібна безперервна робота програми, для відповідної індикації зміни стану перемикачів.

Далі буде розглянуто процес проектування пристрою, зображеного на малюнку 2, з використанням середовища Qsys. Після призначення контактів FPGA, для реалізації з'єднань між паралельними інтерфейсами перемикачів та світлодіодів плати DE1-SoC, проект компілюється. Використовуючи програмне середовище **Intel® FPGA Monitor Program**, конфігурується розроблена схема на пристрої FPGA та завантажується на виконання програма для процесора Nios II.

4. Послідовність виконання роботи

1. Для початку створіть новий проект. Розмістіть його у директорії ***platformdesigner_tutorial*** та назвіть ***lights*** (див. Рис. 3).

Рисунок 3. Створення нового проекту.

Це має бути пустий проект (***empty project***), не додавайте ніяких файлів, виберіть свою мікросхему (**5CSEMA5F31C6** для DE1-SoC) Налаштування **EDA Tool settings** залиште за замовчанням.

2. Відкрийте **Platform Designer Tool**. Для цього оберіть меню ***Tools → Platform Designer***. Відкриється вікно як на рис. 4.

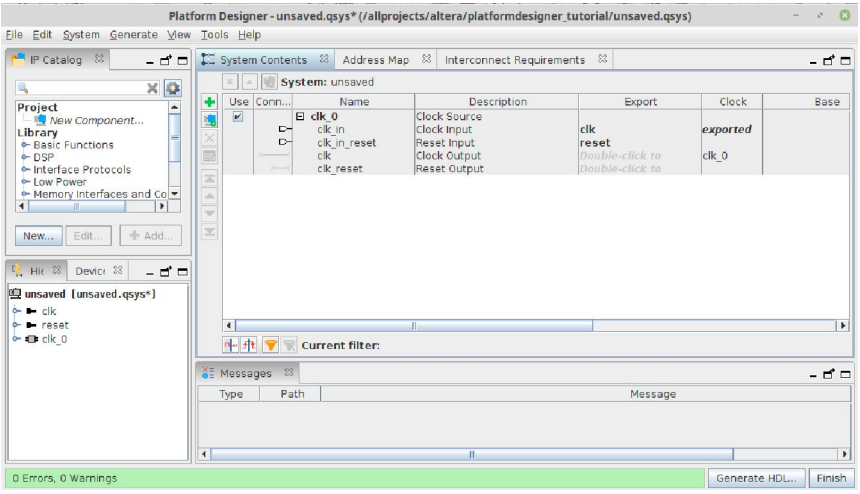


Рисунок 4. Вікно *Platform Designer Tool*

Як ви бачите, система вже містить один компонент ***clk_0***. Двічі натисніть на нього, щоб переглянути його параметри (рис. 5). Частота тактового сигналу зараз встановлена 50 МГц, це можна змінити, або вказати, що частота заздалегідь не відома, але у даній роботі потрібно залишити її саме такою.

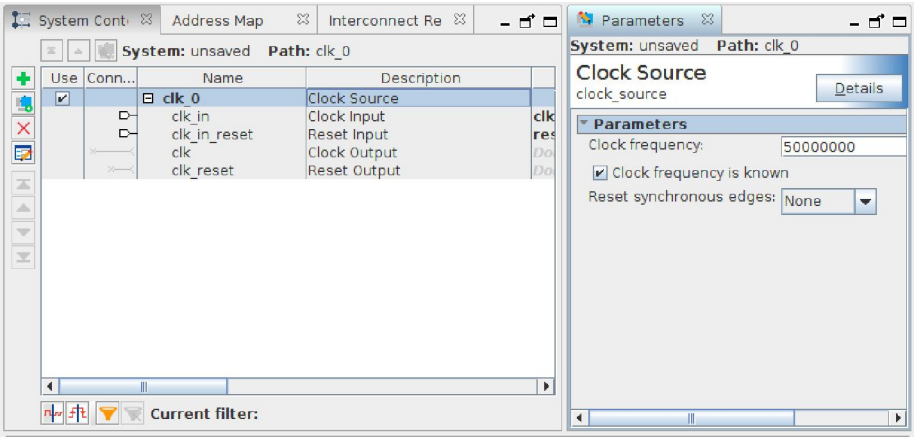


Рисунок 5. Параметри *clk_0*.

3. Додайте до системи процесорне ядро. Для цього, у вкладці ***IP Catalog*** (вкладка знаходиться зліва у вікні ***Platform Designer Tool***) оберіть ***Library → Processors and Peripherals → Embedded Processors → Nios II (Classic) Processor*** та натисніть кнопку ***Add.*** Відкриється вікно зображене на рис. 6.

У консольному вікні (знизу) може з'явитися сповіщення про деякі помилки! Не хвилюйтеся, ви їх пізніше виправите.

4. У вікні налаштування процесорного ядра оберіть ***Nios II/e*** — економічну версію процесора (рис. 7) і натисніть ***Finish***. Усі інші налаштування зробите потім.

На рис. 8 показано, як має виглядати результат.

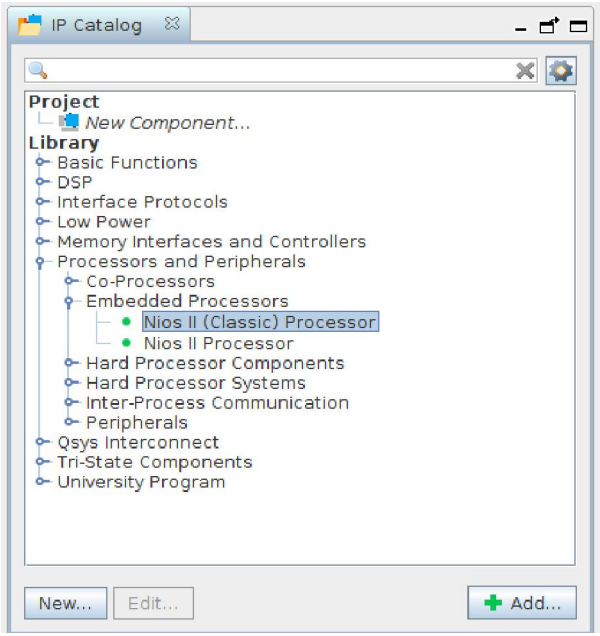


Рисунок 6. Додавання ядра **Nios II**.

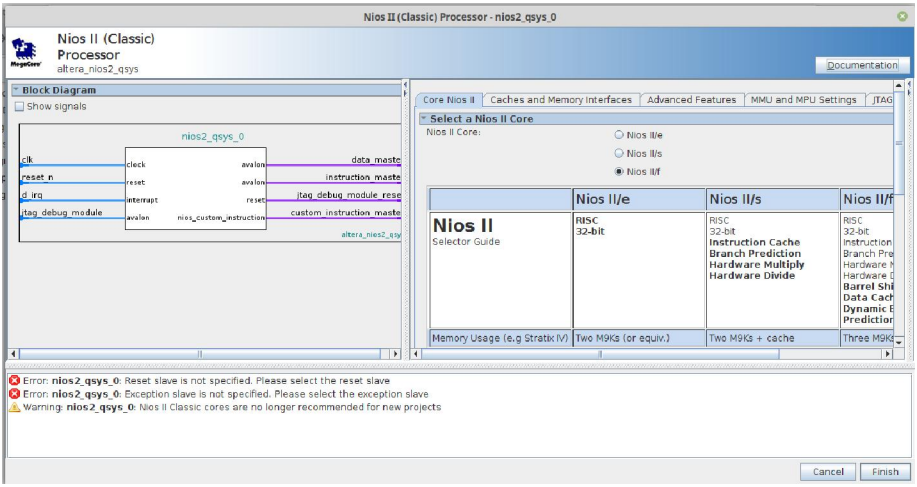


Рисунок 7. Вікно налаштування Nios II.

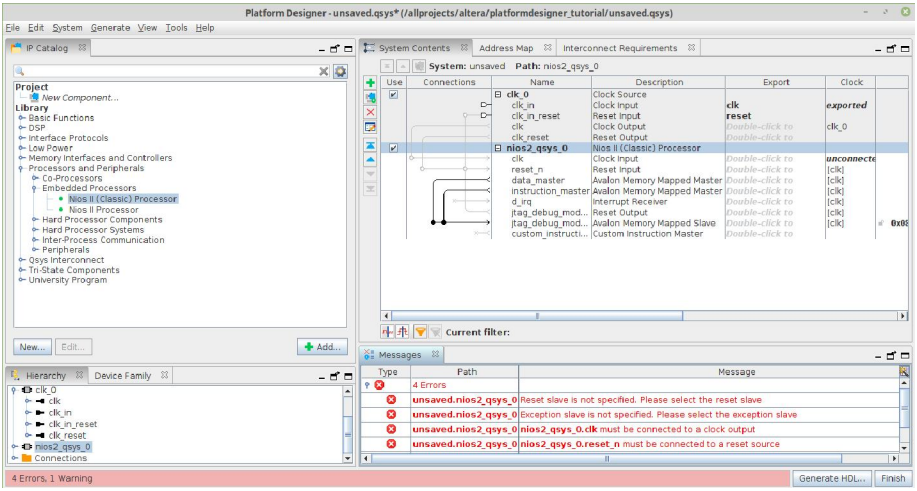


Рисунок 8. Вигляд системи після додавання Nios II/e.

5. Додайте пам’ять. Для цього оберіть ***IP Catalog → Library → Basic Functions → On Chip Memory → On-Chip Memory (RAM or ROM) Intel FPGA IP*** (див. рис. 9).

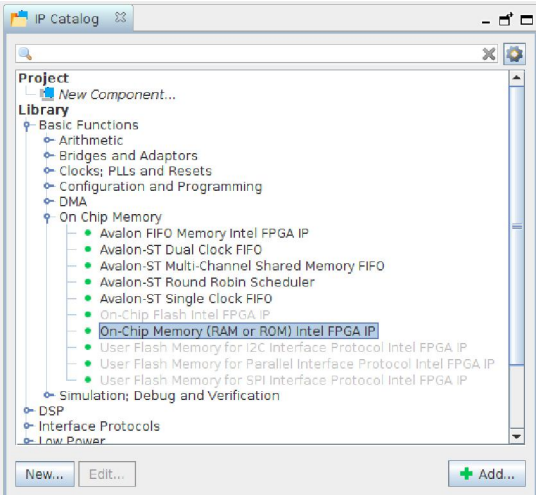


Рисунок 9. Додавання пам’яті.

Зробіть наступні налаштування ***Slave S1 data width = 32*** та ***Total memory size = 4096 bytes*** (рис. 10). Потім натисніть ***Add***.

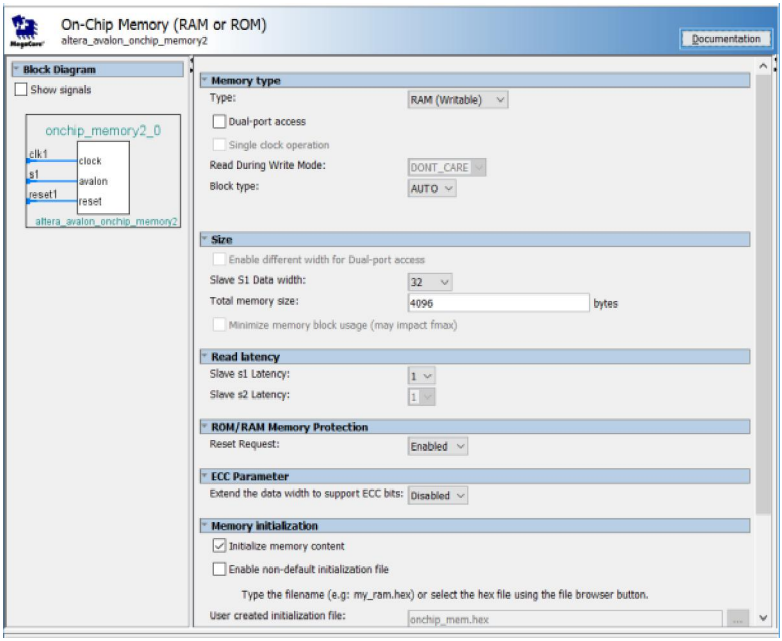


Рисунок 10. Налаштування пам’яті.

6. Додайте два 8-бітні порти введення/виведення. Для цього оберіть ***IP Catalog → Library → Processors and Peripherals→ Peripherals → PIO (Parallel I/O) Intel FPGA IP*** та натисніть ***Add***. Налаштуйте їх так, як зображено на рис.11 та рис.12.

Змініть назви портів введення/виведення на “***switches***” (порт введення) та “***LEDs***” (порт виведення). Для цього потрібно натиснути правою кнопкою миші на назві порту, та обрати команду ***Rename*** з контекстного меню. Потім - ввести нову назву.

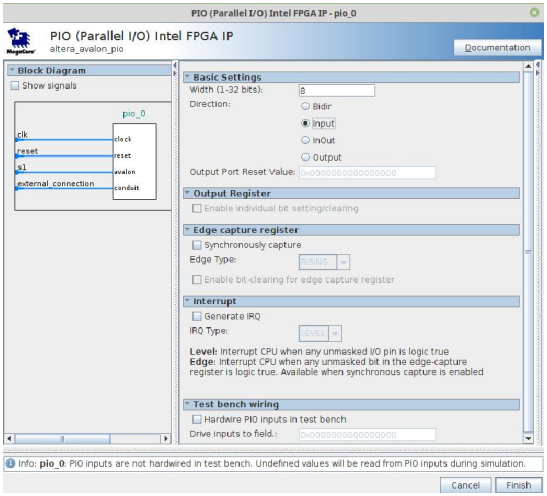


Рисунок 11. Налаштування порту введення.

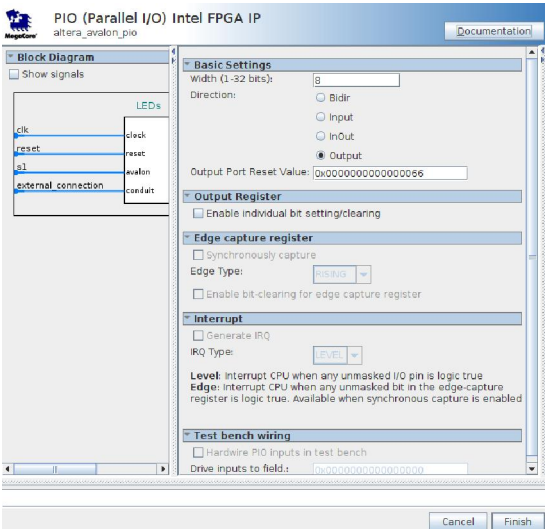


Рисунок 12. Налаштування порту виведення.

7. Додайте JTAG UART, для цього оберіть ***IP Catalog → Library → Interface Protocols → Serial → JTAG UART Intel FPGA IP*** та натисніть ***Add***. Відкриється вікно у якому все залиште за замовчанням.

8. У вікні ***System Contents*** відобразяться додані компоненти, наявні та можливі зв'язки між ними (див. рис. 13).

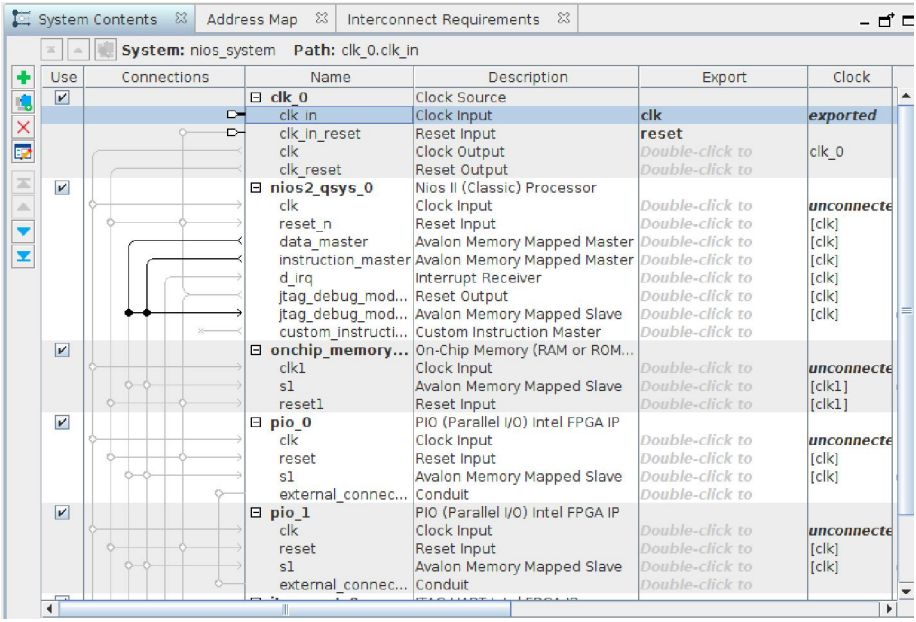


Рисунок 13. Вигляд вікна *System Contents*.

Натискаючи на кружечки у стовпці ***Connections*** можна додавати та прибирати зв’язки між компонентами системи. Налаштуйте зв’язки так, як вказано на рис. 14.

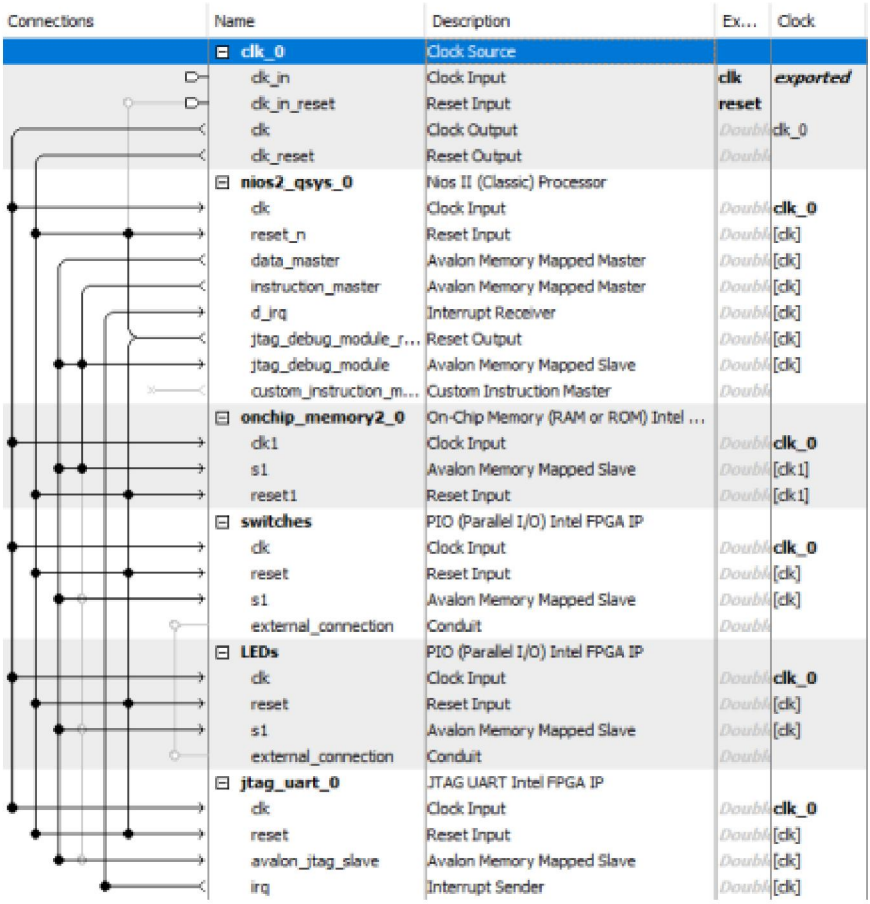


Рисунок 14. Налаштування зв'язків між компонентами.

9. Під'єднайте лінію запиту на переривання (IRQ = Interrupt Request), якщо вона не під'єдналась автоматично до ядра процесора. Призначте перериванню номер 5. Для цього оберіть квадратик у стовпці ***IRQ*** у рядку ***jtag_uart_0 → IRQ*** та введіть "5".

10. Далі необхідно встановити базові адреси для пристроїв, під'єднаних до шини *Avalon*. Їх можна призначити вручну, або автоматично. Блоку пам'яті призначте адресу вручну. Для цього двічі натисніть на значенні у стовпці ***Base*** рядка ***onchip_memory → s1*** (див. рис. 15).

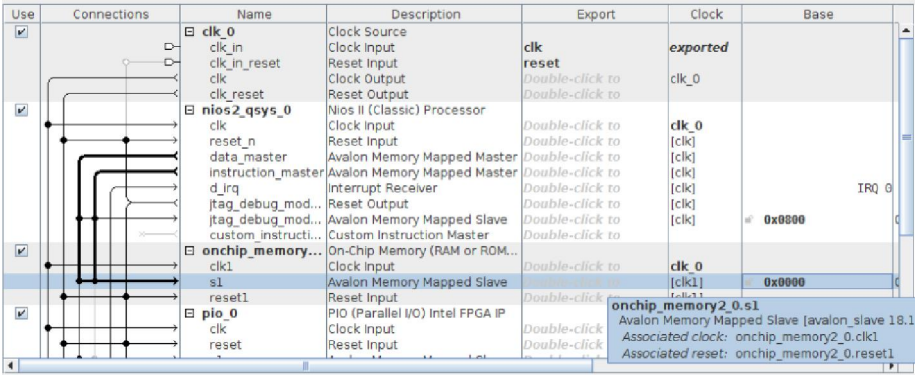


Рисунок 15. Призначення адреси блоку пам’яті.

Встановіть значення 0x0000. Значення має бути саме таким — це важливо! Потім заблокуйте цю адресу, натиснувши на відповідний символ поруч.

11. Інші адреси призначте автоматично. Оберіть у меню команду ***System → Assign Base Addresses***. Наразі, базові адреси компонентів визначені коректно.

12. На наступному кроці необхідно виконати налаштування у пам’яті адреси розташування векторів переривань (***Exception memory***) та початку виконання програми (***Reset Vector memory***) для ядра Nios II. Для цього натисніть правою кнопкою миші на ***nios2_qsys_0*** та оберіть з контекстного меню команду ***Edit***. Для параметрів ***Reset vector memory*** та ***Exception vector memory*** оберіть ***onchip_memory2_0.s1*** (див. рис. 16).

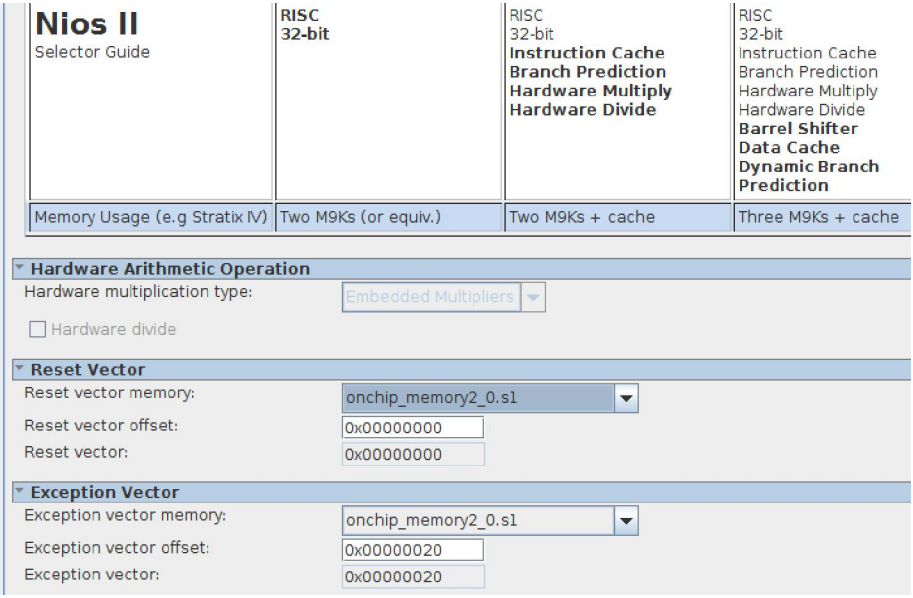


Рисунок 16. Налаштування векторів переривань та початку виконання програми процесора Nios II.

Всі інші параметри залиште за замовченням. Натисніть ***Finish***. Зверніть увагу - всі помилки (Errors) зникли!

13. Далі необхідно налаштувати передачу сигналів портів ***switches*** та ***LEDs*** за межі системи, яка налаштовується у Platform Designer Tool. Для цього, двічі натисніть на напис ***Double-click to export*** у стовпці ***Export*** у рядку ***switches → external_connection*** (див. рис. 17). Впишіть ім'я ***switches***. Так само для порту ***LEDs*** впишіть ім'я ***leds***. Результат зображено на рис. 18.

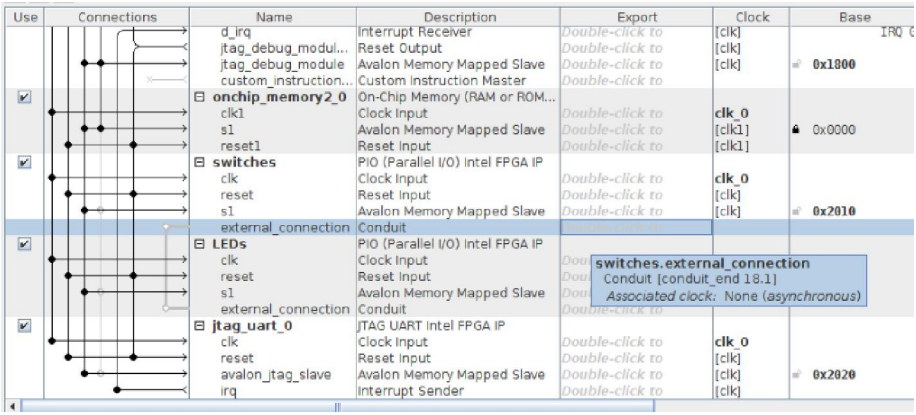


Рисунок 17. Натисніть на напис *Double-click to export.*

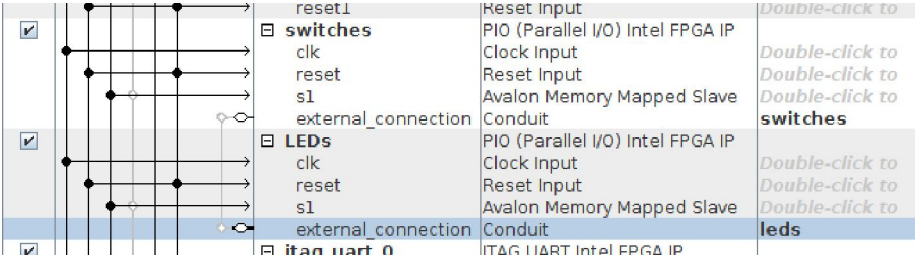


Рисунок 18. Налаштування портів.

14. Збережіть систему під назвою ***nios_system***.

15. Далі оберіть команду ***Generate → Generate HDL***. Для ***Create simulation model*** оберіть ***None***. У цій роботі не буде виконуватися симуляція пристрою. Результати налаштувань відображені на рис. 19. Натисніть *Generate* та дочекайтеся сповіщення ***Generate completed***.

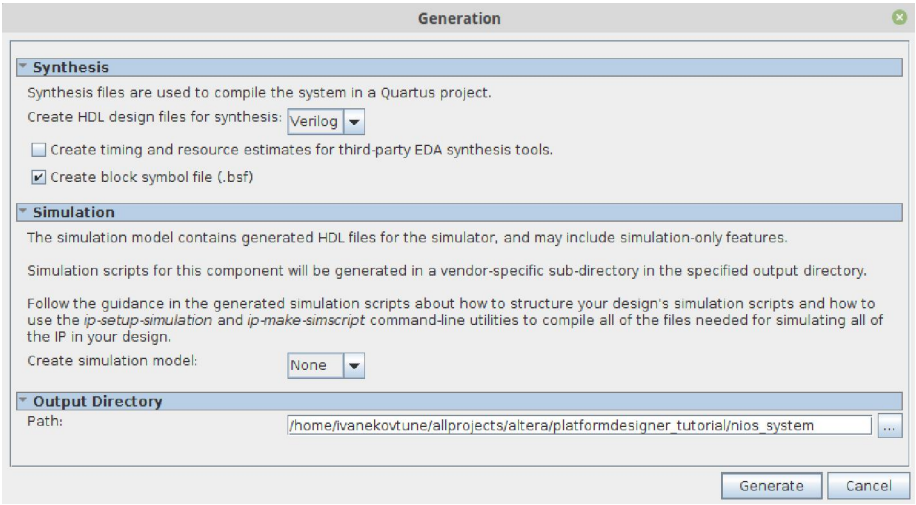


Рисунок 19. Налаштування у вікні *Generation*.

16. Поверніться до головного вікна середовища ***Quartus Prime***. Наразі необхідно долучити до проекту створену ***nios_system***. Для цього у вікні ***Project Navigator → Files*** (зліва) натисніть правою кнопкою миші на піктограмі ***Files*** та оберіть команду ***Add/Remove Files in Project*** з контекстного меню. Відкриється вікно додавання файлів. Додайте файл ***platformdesigner_tutorial/nios_system/synthesis/nios_system.v*** (рис. 20). Так само додайте файл ***nios_system.qip***, що знаходиться у папці разом з ***nios_system.v***.

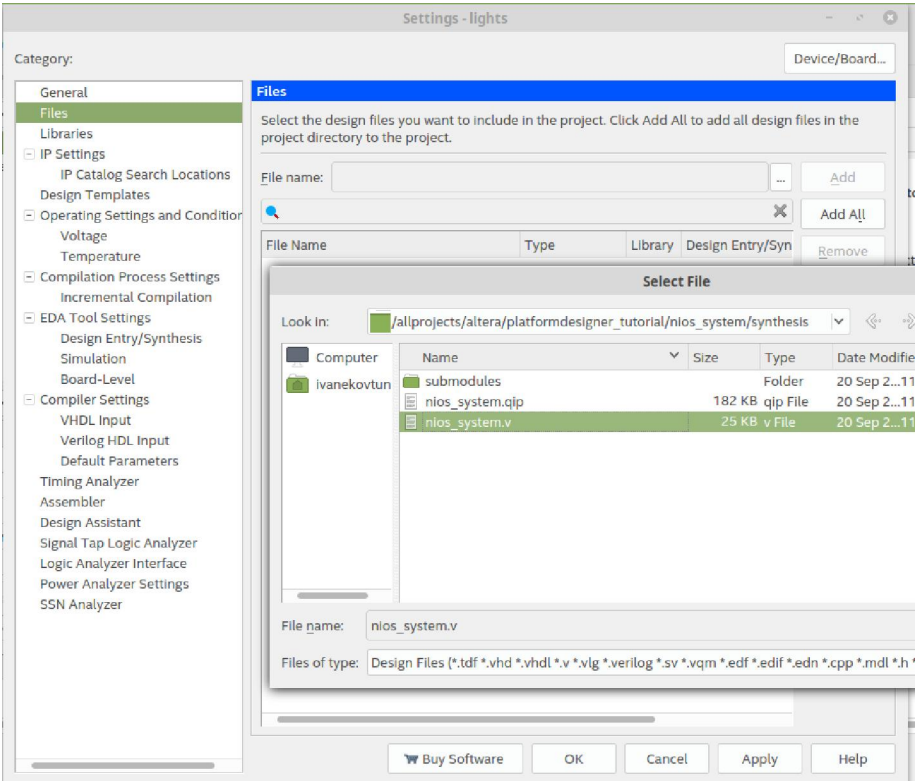


Рисунок 20. Додання файлу *nios_system.v*

17. Створіть файл верхнього рівня ієрархії ***lights.v***. У ньому буде підключено щойно згенеровану та додану до проекту систему з ядром Nios. Додайте у файл ***lights.v*** наступний текст:

```
module lights (CLOCK_50, SW, KEY, LEDR);
    input CLOCK_50;
    input [7:0] SW;
    input [0:0] KEY;
    output [7:0] LEDR;

// Instantiate the Nios II system module generated
// by the Platform Designer tool:
```

```
    nios_system NiosII (
        .clk_clk(CLOCK_50),
        .reset_reset_n(KEY),
        .switches_export(SW),
        .leds_export(LEDR));

endmodule
```

18. Виконайте призначення контактів мікросхеми (pin assignment) для плати DE1-SoC. Для цього оберіть меню ***Assignments → Import Assignments*** та вкажіть файл ***DE1_SoC.qsf***.

19. Запустіть компіляцію проекту.

20. Завантажте скомпільований проект до плати DE1-SoC за допомогою ***Programmer Tool***. (Підказка: якщо ви раніше не працювали саме з DE1-SoC, то вам потрібно буде натиснути ***Add Device*** та обрати ***Soc Series V → SOCVHPS*** перш ніж завантажувати).

21. Наступний етап – розробка програмного забезпечення для процесорного ядра. Створіть файл ***lights.s*** у директорії ***platformdesigner_tutorial/app_software*** (директорію теж необхідно створити). Помістіть у файл наступний текст:

```
.equ    switches,   0x00002010
.equ    leds,       0x00002000
.global _start
_start: movia   r2, switches
        movia   r3, leds
LOOP:   ldbio   r4, 0(r2)
        stbio   r4, 0(r3)
        br      LOOP
.end
```

Це програма мовою Асемблер для Nios II. Замість чисел 0x00002010 та 0x00002000 укажіть базові адреси відповідних портів вашого пристрою (якщо адреси відрізняються).

22. Запустіть ***Intel FPGA Monitor Program***. (Підказка: ця програма не є частиною Quartus Prime, встановлюється і запускається окремо від нього).

23. Коли відкриється головне вікно програми, оберіть команду ***File → New Project***. На першій вкладниці вкажіть наступні значення:

Project Directory = platformdesigner_tutorial/app_software

Project Name = lights_example

Architecture = Nios II

Натисніть ***Next.***

24. На вкладинці ***Specify the system*** вкажіть наступні параметри (рис. 21):

Select a system = <Custom system>

System description file =
platformdesigner_tutorial/nios_system.sopcinfo

FPGA programming (SOF) file =
platformdesigner_tutorial/output_files/lights.sof

Preloader = Not required

Натисніть ***Next.***

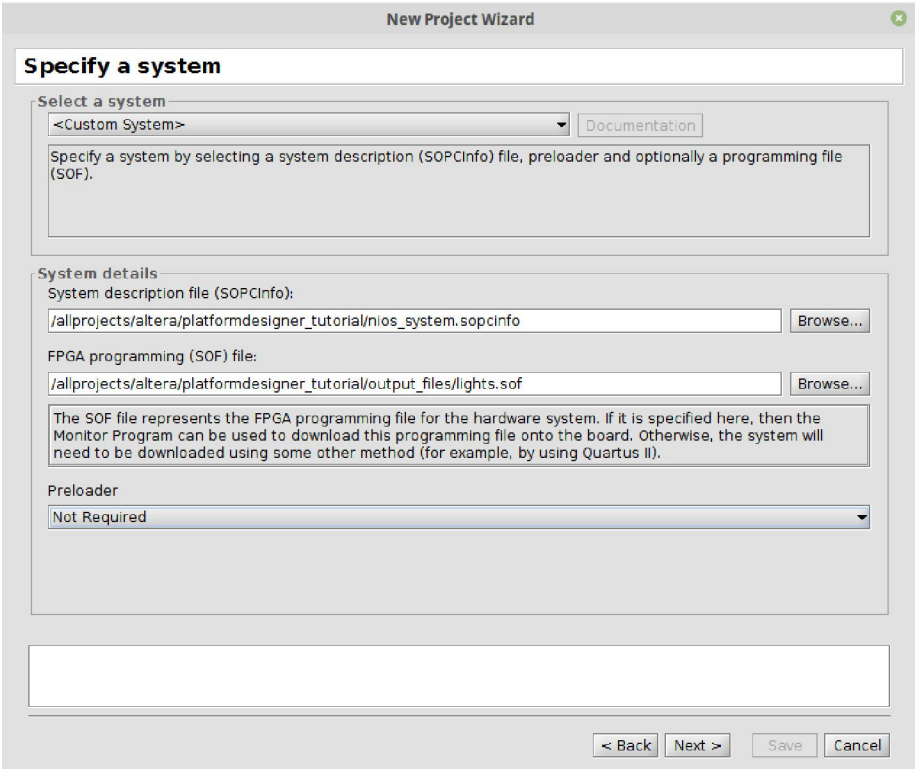


Рисунок 21. Вкладинка ***Specify the system***.

25. На вкладинці ***Specify a program type*** укажіть значення ***Program Type = Assembly Program***. Натисніть ***Next.***

26. На вкладинці ***Specify program details*** натисніть ***Add,*** та оберіть файл програми ***lights.s***. Вкажіть ***Start symbol = _start***. Натисніть ***Next.***

27. На вкладинці ***Specify system parameters*** укажіть наступні значення

Host connection = DE-SoC [3-2]

Processor = nios2_qsys_0

Terminal device = jtag_uart_0

(Підказка: якщо немає параметра для ***Host connection***, то підключіть плату і натисніть ***Refresh***).

28. На останній вкладинці (рис. 22) натисніть ***Finish***, або ***Save***.

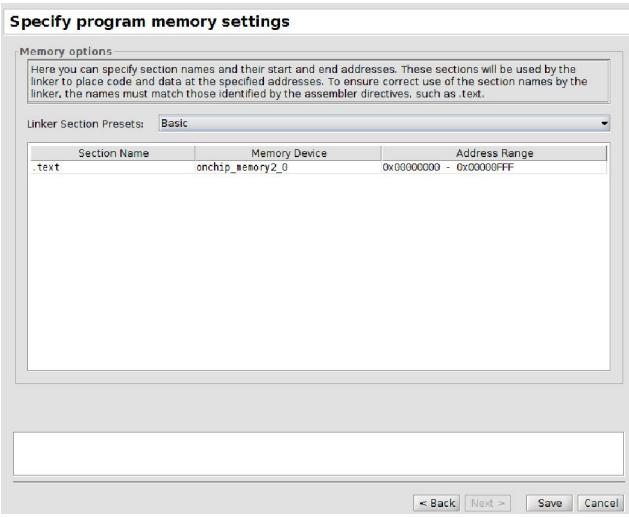


Рисунок 22. Вкладинка ***Specify program memory settings***

На запит чи завантажувати систему на плату (рис. 23) відповідайте ***Yes***.

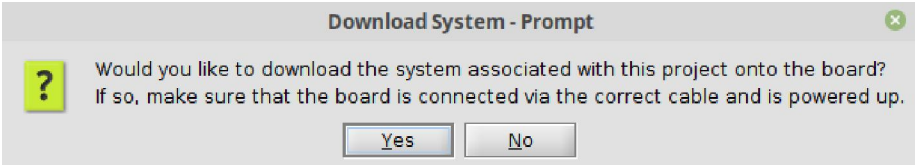


Рисунок 23. Запит, на який треба відповісти ***Yes***.

29. Після сповіщення про успішне завантаження, у вікні ***Intel FPGA Monitor Program — lights_example*** (рис. 24) оберіть меню ***Actions → Compile & Load.***

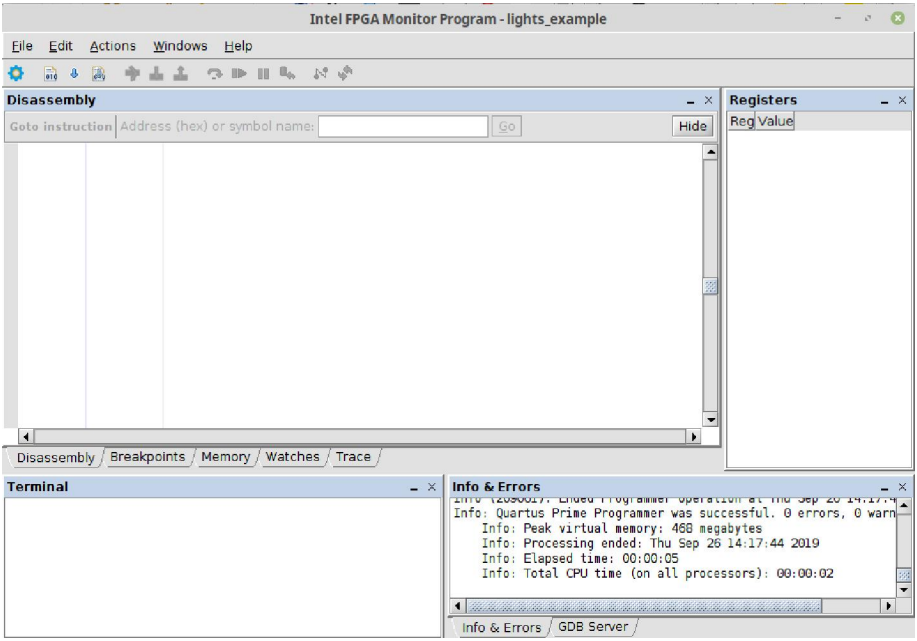


Рисунок 24. Вікно ***Intel FPGA Monitor Program***.

Звертаємо увагу! Якщо ви отримали помилку та у вікні ***Info & Errors*** (рис. 25) з'явився напис (це стосується Linux):

make: *** No rule to make target 'compile'. Stop.

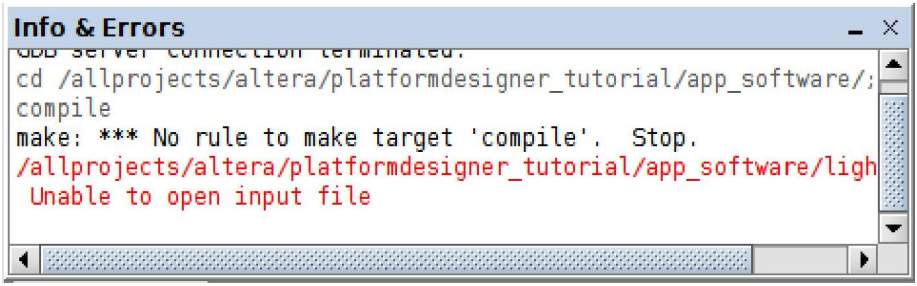


Рисунок 24. Вікно ***Info & Errors***.

то виправити помилку можливо, якщо у директорії ***platformdesigner_tutorial/app_software*** відкрити файл ***makefile,*** та у строчці, наступній від ***# Targets***, слово ***COMPILE*** (великими літерами) замініте на ***compile*** (маленькими літерами).

30. Після завантаження програми, відкриється вікно дебагера. Воно показано на рис. 26.

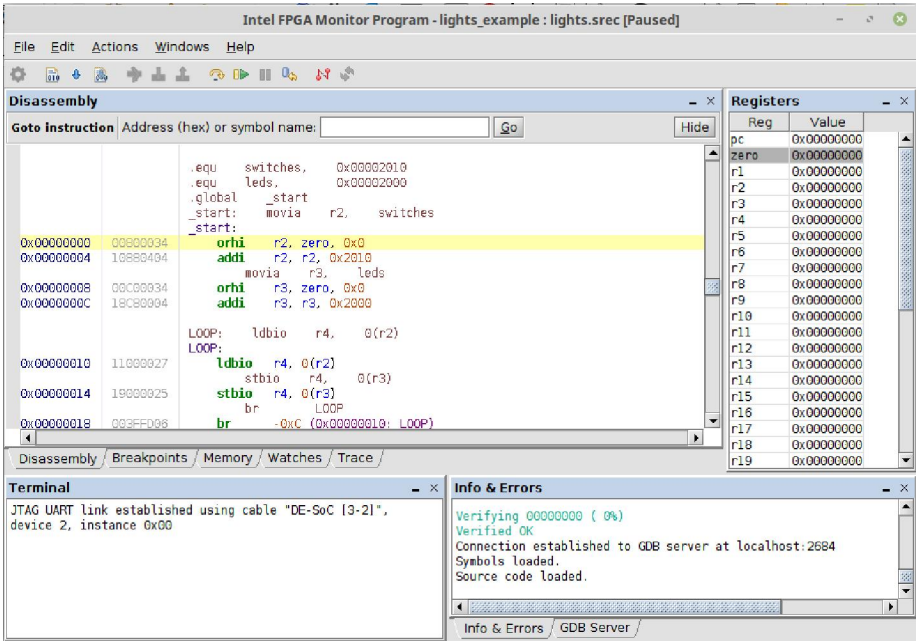


Рисунок 26. Вікно дебагера.

31. Запустіть програму на виконання у безперервному (зелений трикутник), або покроковому режимі (жовта стрілочка) та спостерігайте її виконання.

Переконайтеся, що червоні світлодіоди змінюють свій стан відповідно до стану перемикачів при виконанні інструкції ***stbio R4, 0(R3).***

5. Самостійна робота

Завдання 1. Для синтезованого процесорного ядра розробіть програму, яка використовує більш складну залежність між комбінацією перемикачів та послідовністю включення світлодіодів (наприклад, «вогник що біжить» та ін.). У якості шаблону використовуйте наступний код мовою С:

```
#define switches (volatile char *) 0x0002010
#define leds (char *) 0x0002000
void main()
{       while (1)
        *leds = *switches;
}
```

Завдання 2. Створіть новий проект та перевірте роботу програми, що обчислює результат накопичення добутків двох векторів.

$$\text{Dot product} = \Sigma A(i) \times B(i)$$

Код програми наведено нижче. Поясніть, як працює програма та перевірте її виконання у покроковому режимі.

```
.include "nios_macros.s"
.global _start
_start:
    movia r2, AVECTOR    /* Register r2 is a pointer to vector A */
    movia r3, BVECTOR   /* Register r3 is a pointer to vector B */
    movia r4, N
    ldw r4, 0(r4) /*Register r4 is used as the counter for loop iterations*/
    add r5, r0, r0  /* Register r5 is used to accumulate the product */
LOOP:   ldw r6, 0(r2)      /* Load the next element of vector A */
        ldw r7, 0(r3)      /* Load the next element of vector B */
```

```
    mul r8, r6, r7      /* Compute the product of next pair of elements */
    add r5, r5, r8      /* Add to the sum */
    addi r2, r2, 4      /* Increment the pointer to vector A */
    addi r3, r3, 4      /* Increment the pointer to vector B */
    subi r4, r4, 1      /* Decrement the counter */
    bgt r4, r0, LOOP        /* Loop again if not finished */
    stw r5, DOT_PRODUCT(r0)       /* Store the result in memory */
STOP: br STOP
N:
.word 6                 /* Specify the number of elements */
AVECTOR:
.word 5, 3, -6, 19, 8, 12         /* Specify the elements of vector A */
BVECTOR:
.word 2, 14, -3, 2, -5, 36        /* Specify the elements of vector B */
DOT_PRODUCT:
.skip 4
```

Важливе зауваження!!!! *Операція перемноження (mul) підтримується процесорним ядром NiosII тільки версій* ***standard*** *або* ***fast****.*